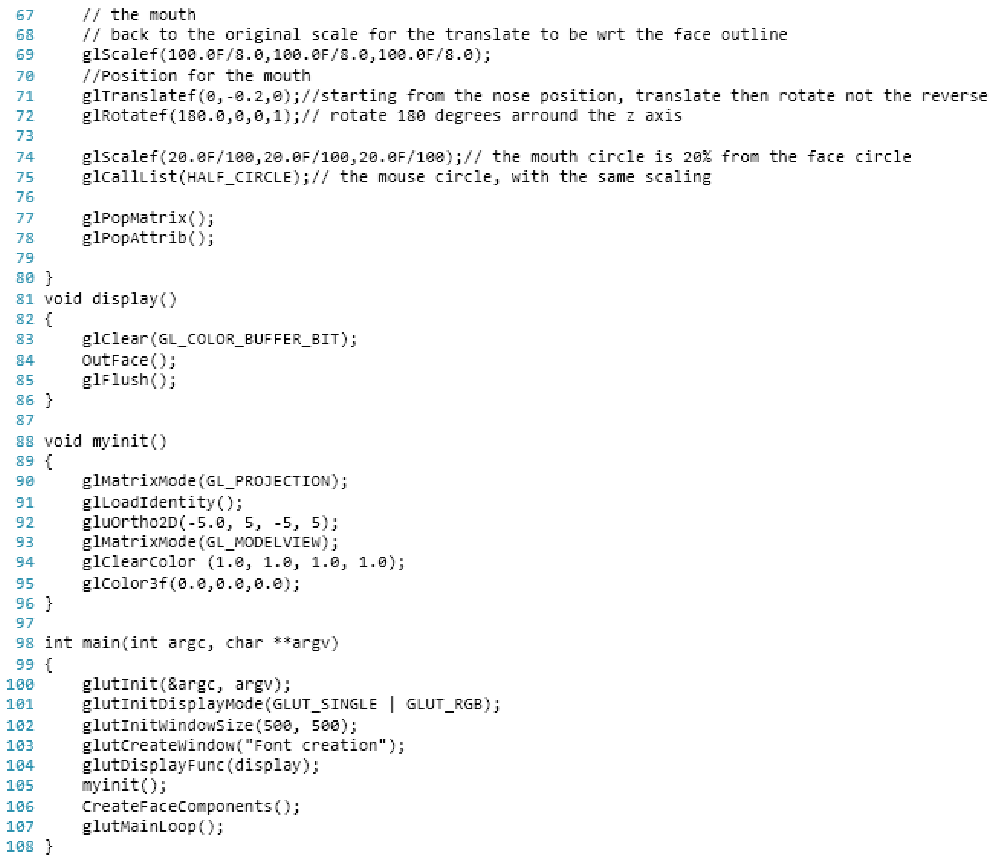
Sheet 8 solution

1. In hierarchical modeling, objects are specified using other objects. For example, an object A is specified relative to its center which is usually assumed to be located at the origin (object coordinate system). Object A can then be used in another object (say B) specification. To put A in its right place as a part of B, the required transformations are done. Other object C may be specified using B (which in turn contains A) and so on.

   The advantages of hierarchical modeling
   - It motivates modularity is design. This simplifies the design and make its modification more easy and structured
   - A components is constructed once and is used any number of time. In OpenGL this can be implemented using display lists, hence leads to performance enhancement.

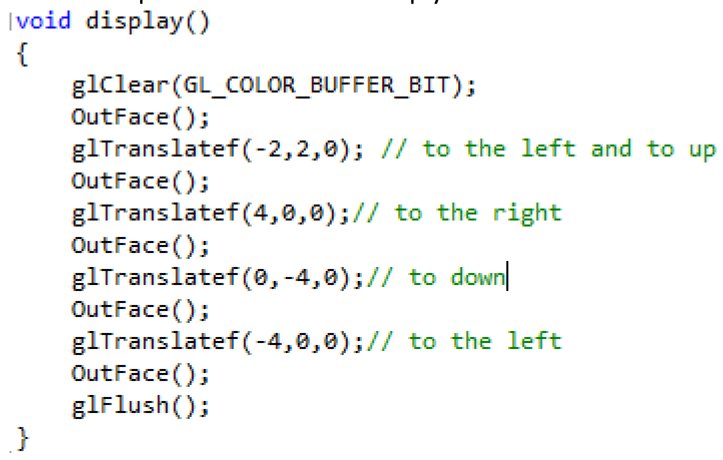
2.

```
#include "stdafx.h"
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <string.h>

#define CIRCLE 1 // for circle display list
#define HALF_CIRCLE 2 // for circle display list
// create the face components  (the component box is 1 by 1)
void CreateFaceComponents()
{
    // define the circle display list
    // The circle is 1 by 1 centered at the origin
    float angle;
    glNewList(CIRCLE, GL_COMPILE);
    glBegin(GL_LINE_LOOP);
    for(int i=0;i<36;i++)// each 10 degrees angle
    {
        angle=3.14159/18*i;// 10 defrees is radians
        glVertex2f(cos(angle),sin(angle));
    }
    glEnd();
    glEndList();
    // define the half circle display list
    // The circle is 1 by 1 centered at the origin
    glNewList(HALF_CIRCLE, GL_COMPILE);
    glBegin(GL_LINE_STRIP);
    for(int i=0;i<=18;i++)// each 10 degrees angle
    {
        angle=3.14159/18*i;// 10 defrees is radians
        glVertex2f(cos(angle),sin(angle));
    }
    glEnd();
    glEndList();
}
void OutFace()
{
    // this function play with the transformation
    glPushAttrib(GL_ALL_ATTRIB_BITS);
    glPushMatrix();

    // the outer outline
    glCallList(CIRCLE);

    // the left eye
    // position for the left eye
    glTranslatef(-0.25,0.25,0);// starting from the center of the outer ouline circle
    glScalef(8.0F/100,8.0F/100,8.0F/100);// the eye circle is 8% from the face circle
    glCallList(CIRCLE);

    // the right eye
    // back to the original scale for the translate to be wrt the face outline
    glScalef(100.0F/8.0,100.0F/8.0,100.0F/8.0);
    // position for the right eye starting
    glTranslatef(0.5,0,0);// starting from the center of the left eye  circle
    glScalef(8.0F/100,8.0F/100,8.0F/100);// the eye circle is 8% from the face circle
    glCallList(CIRCLE);// the right eye circle, with the same scaling

    // the nose
    // back to the original scale for the translate to be wrt the face outline
    glScalef(100.0F/8.0,100.0F/8.0,100.0F/8.0);
    //Position for the nose
    glTranslatef(-0.25,-0.5,0);//starting from the right eye position
    glScalef(8.0F/100,8.0F/100,8.0F/100);// the nose circle is 8% from the face circle
    glCallList(CIRCLE);// the mouse circle, with the same scaling

```



```
    // the mouth
    // back to the original scale for the translate to be wrt the face outline
    glScalef(100.0F/8.0,100.0F/8.0,100.0F/8.0);
    //Position for the mouth
    glTranslatef(0,-0.2,0);//starting from the nose position, translate then rotate not the reverse
    glRotatef(180.0,0,0,1);// rotate 180 degrees arround the z axis

    glScalef(20.0F/100,20.0F/100,20.0F/100);// the mouth circle is 20% from the face circle
    glCallList(HALF_CIRCLE);// the mouse circle, with the same scaling

    glPopMatrix();
    glPopAttrib();

}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    OutFace();
    glFlush();
}

void myinit()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-5.0, 5, -5, 5);
    glMatrixMode(GL_MODELVIEW);
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glColor3f(0.0,0.0,0.0);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Font creation");
    glutDisplayFunc(display);
    myinit();
    CreateFaceComponents();
    glutMainLoop();
}
```

3. The solution is the same as in problem 2 but the disaply function is modefied as follows:

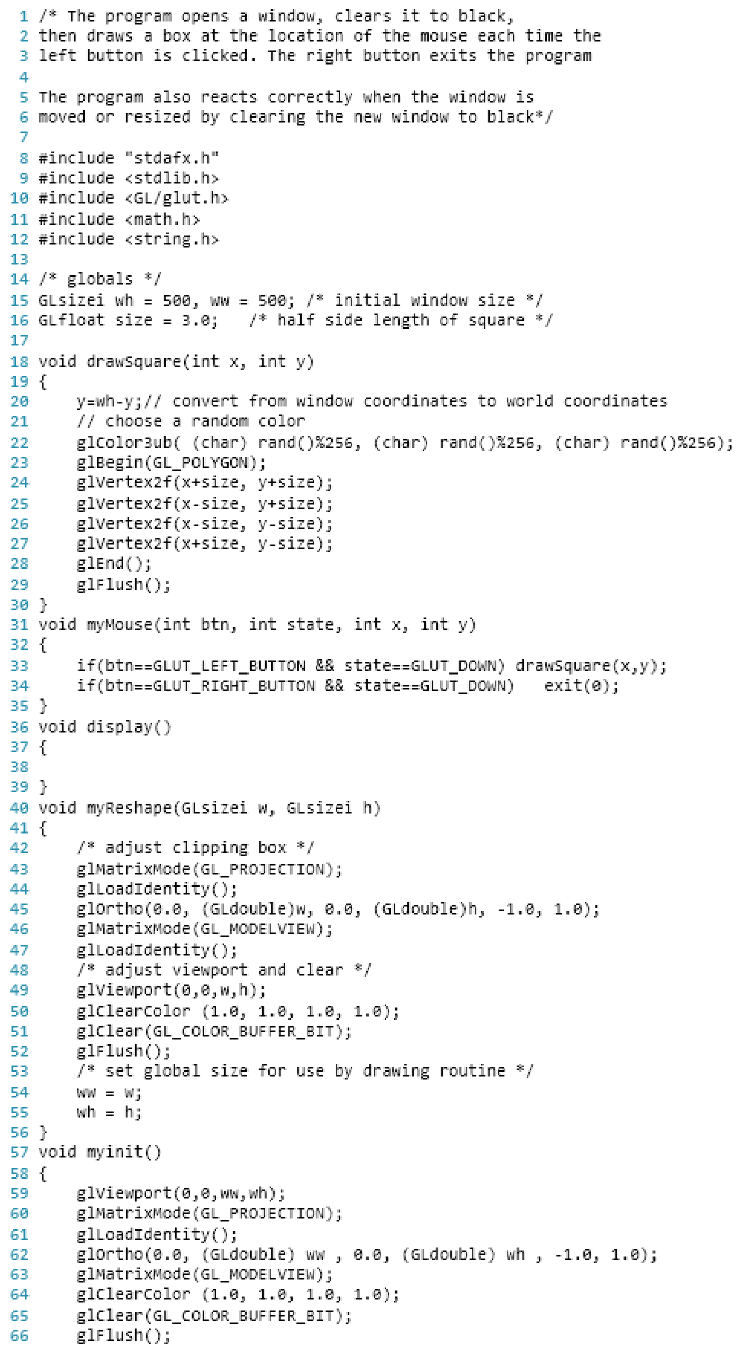


```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    OutFace();
    glTranslatef(-2,2,0); // to the left and to up
    OutFace();
    glTranslatef(4,0,0);// to the right
    OutFace();
    glTranslatef(0,-4,0);// to down
    OutFace();
    glTranslatef(-4,0,0);// to the left
    OutFace();
    glFlush();
}
```

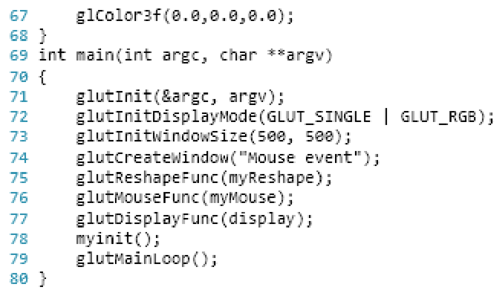
4. **glutReshapeFunc**(…): Is used to register a callback function for the reshape event. The reshape event occurs when the drawing window size is changed, for example as a result of user interaction.
   glutMotionFunc(): Is used to register a callback function for the motion event. The motion event

occurs when the user does an active motion with a pointing device. Active motion in the case of a mouse means that the mouse pointer moves while a button is pressed; a passive motion is the movement of the cursor without pressing any buttons.
glutMouseFunc(): Is used to register a callback function for the mouse event. The mouse event occurs when one of the mouse buttons changes state (pressed or released).

5.



```
/* The program opens a window, clears it to black,
then draws a box at the location of the mouse each time the
left button is clicked. The right button exits the program

The program also reacts correctly when the window is
moved or resized by clearing the new window to black*/

#include "stdafx.h"
#include <stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include <string.h>

/* globals */
GLsizei wh = 500, ww = 500; /* initial window size */
GLfloat size = 3.0;   /* half side length of square */

void drawSquare(int x, int y)
{
    y=wh-y;// convert from window coordinates to world coordinates
    // choose a random color
    glColor3ub( (char) rand()%256, (char) rand()%256, (char) rand()%256);
    glBegin(GL_POLYGON);
    glVertex2f(x+size, y+size);
    glVertex2f(x-size, y+size);
    glVertex2f(x-size, y-size);
    glVertex2f(x+size, y-size);
    glEnd();
    glFlush();
}
void myMouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN) drawSquare(x,y);
    if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)   exit(0);
}
void display()
{

}
void myReshape(GLsizei w, GLsizei h)
{
    /* adjust clipping box */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, (GLdouble)w, 0.0, (GLdouble)h, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    /* adjust viewport and clear */
    glViewport(0,0,w,h);
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
    /* set global size for use by drawing routine */
    ww = w;
    wh = h;
}
void myinit()
{
    glViewport(0,0,ww,wh);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, (GLdouble) ww , 0.0, (GLdouble) wh , -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glFlush();
```



```
67     glColor3f(0.0,0.0,0.0);
68 }
69 int main(int argc, char **argv)
70 {
71     glutInit(&argc, argv);
72     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
73     glutInitWindowSize(500, 500);
74     glutCreateWindow("Mouse event");
75     glutReshapeFunc(myReshape);
76     glutMouseFunc(myMouse);
77     glutDisplayFunc(display);
78     myinit();
79     glutMainLoop();
80 }
```

6. The solution is the same as the solution of problem 5 except the addition of the statement
   `glutMotionFunc(drawSquare);`
   direclly after the line no. 76
